Retroviruses - how viruses fight back

This paper was presented at Virus Bulletin 1994 Conference in Jersey.

Mikko Hyppönen
F-Secure Ltd

June 1994

"The GoldBug virus has extensive anti-anti-virus routines. It can install itself while several resident anti-virus monitors are running. It will prohibit most popular anti-virus programs from running, and will also by-pass several integrity checking programs"
-from the original source code of the GoldBug virus

Abstract

This paper will discuss methods viruses use or might use in the future to attack anti-virus programs. Attacks of this kind are becoming more common, as virus writers seem to be constantly looking for ways to make their viruses more efficient and vigorous. This paper also suggests how to make antivirus products more resistant against such attacks. The scope of this paper is limited to PC compatible machines.

1. Introduction

There is a constant battle going on between computer virus authors and virus fighters. Virus writers are looking for methods to create more complicated, more difficult-to-analyse and more inconspicuous viruses. At the same time the anti-virus people are building methods to address these threats.

It's not surprising that virus authors have realised that anti-virus tools are one of the worst enemies for their creations. The logical step for them was to make ...their viruses fight back, either directly or indirectly..

Several viruses explicitly target anti-virus programs. The attack routines might be generic or targeted against a specific program. Obviously many virus authors consider attack to be the best defence, when the objective is to keep the virus alive in order to spread as widely as possible.

There is a battle going on in computer systems world-wide - it's survival of the fittest, one might say. Hopefully this paper will provide some ideas to make anti-virus applications fitter than the viruses.

2. A virus that fights back

For the purpose of this paper, a retrovirus is defined as follows:

Retrovirus is a computer virus that specifically tries to by-pass or hinder the operation of an antivirus program or programs. The attack may be specific to a known product or a generic one.

Retroviruses are sometimes known as anti-anti-viruses. Anti-anti-viruses should not be confused with anti-virus-viruses, which are viruses that will disable or disinfect other viruses. To avoid confusion, the term retrovirus will be used here.

The creation of a virus which has retro-routines included is not necessarily a difficult task. In most cases, the virus writers have access to the anti-virus programs they want to by-pass. All they need to do is to experiment by trial and error until they find a way to attack the anti-virus program in a way that the anti-virus developer has not foreseen. [Siilasmaa]

Some virus authors have gone all the way and disassembled the offending anti-virus programs in order to find the most effective way to attack it. They often look for methods to attack a product in such a way that it would be most difficult to circumvent in future versions of the product.

As the virus authors are pretty efficiently connected to each others via different types of electronic networks, information on how to attack specific products spreads quickly.

It should be noted that the virus writers typically have access to only those antivirus products that are available as freeware or shareware. Some virus exchange BBS systems are known to make pirated copies of commercial products available, but the shareware products seem to be targeted most [Fellows].

It can be expected that more retroviruses, using more advanced retro-routines will be seen in the future.

3. Rules of the game

Viruses using retro-routines started to show up during late 1980's - before that there was no point in creating retroviruses, as anti-virus products weren't widely used. As anti-virus programs have increased in popularity, more viruses attempt to subvert them in some way.

Several approaches are possible, including:

- modifying the code of an anti-virus program file or the image in memory
- detecting when an anti-virus program is activating, and either hiding itself, stopping the execution of the program or triggering a destructive routine
- altering the computing environment in a way that affects the operation of an anti-virus
- program using methods in the virus code that cause problems for anti-virus programs
- exploiting a specific weakness or a backdoor in an anti-virus program
- using generic methods that generally make it difficult or potentially dangerous to detect, identify or disinfect the virus

The basic principle is that the virus must somehow hinder the operation of an anti-virus program in such a way that the virus itself benefits from it.

Methods like encryption, stealth, polymorphic routines, code armouring, anti-debugging tricks and confusion code can also be considered as an attack against anti-virus programs. However, they are often generic in type and are outside the scope of this paper.

4. Attacks against non-resident scanners

Non-resident scanners are probably the most commonly used anti-viral products. They are also the favourite target of real-world retroviruses.

There are several different ways a scanner can be attacked against.

4.1 Deletion and replacement

A virus can locate the anti-virus program and delete it. A more sophisticated attack would be a modification or a patch that would alter the operation of the scanner in a way that would be beneficial to the virus. A virus could locate the search strings used by the scanner and overwrite them, making the scanner unable to find any virus, but still appear to be functional.

A virus could replace the scanner program with a Trojan horse which could trigger a damage routine when run or just simply display an error message and abort. Such an error message would also make the scanning product look bad in the eyes of the users, especially if the error message would be something like 'only 620kB of free DOS memory, unable to run' or 'BRUN30 GW-Basic run-time library not found, aborting'.

If the virus stays resident in memory, it can do similar attacks when it sees that the anti-virus program is executed. It can also by-pass a self-check routine of an anti-virus program by patching it only after the application has finished the check on its own code.

4.2 Modification of parameters

There is at least one known case of a virus that modifies the command-line parameters when it sees a specific anti-virus program to be started (see below). Such technique would allow the virus to modify the operation of the scanner to its advantage without patching the actual program code.

A similar attack might be possible by modifying the configuration file of an anti-virus program - these are often left unencrypted and are not checked for such modifications.

4.3 Altering the output

If the visual interface of the anti-virus program isn't complex (ie. command-line driven), it might be feasible for a retro-virus to mimic the operation of the program. This way, the user might not notice anything strange.

A variation of the theme would be that the virus would patch the texts displayed by the product. If the text string 'Virus found!" were to be changed to 'All clear!', a typical user wouldn't probably doubt anything.

In many installations, anti-virus programs are run automatically and the alarms are set off depending on the exit codes (errorlevels) returned by a program. A successful attack in such a system might consist of a retrovirus that would always set the return-code of an anti-virus program to zero.

4.4 False false alarms

Scanners are also prone to false alarms ie. detecting a virus in a clean file. Viruses can use this as one way to attack. If a virus incorporates code sections from popular applications, it is quite possible that an anti-virus vendor without a proper false-positive testing routine might include a search string that would cause a large amount of false positives.

One way to implement this kind of attack would be to include an encryption routine to a virus, but borrow the decryption code from some known application - the encryption would limit the traditional search strings to only strings that would cause false positives, and this in itself would cause problems for some scanning products.

4.5 Problems with packed files

Several scanners are able to scan inside compressed executables that have been packed with some of the most popular EXE-packers. Some scanners do not scan packed files at all, but only flag them as packed so the user is aware of them. This provides one way a virus could cause problems for a scanner. If a virus would use a section of fake code that would make an infected program look like it has been packed, it would by-pass the scanning by such a product completely. The virus could also replicate in packed form, making it even more difficult to detect by some scanners.

A similar attack might be possible against products that actually unpack the programs and scan underneath the packing. In order to uncompress the program, the scanner fetches program info from the unpacking code. If this code would contain irrational values, it could cause some scanners to crash or run out of memory.

4.6 One man's data is another man's code

Almost all scanners default to scanning only the executable files instead of all files. File type is usually determined by the extension (ie. COM, EXE, SYS).

Since a virus can control the system in any way it wants, one way to by-pass a scanner would be to change the file-extension of every infected file to a non-executable one, for example from EXE to XEX. While the virus is resident in memory, it would use stealth techniques to hide this change - but would make sure that all executables copied to floppies would contain to valid extension to ensure the virus gets a chance to spread. The advantage of such a method would be that even if the machine was booted up from a clean diskette and all executables would be scanned with a scanner that could detect the virus, it would only be found in the inital carrier file.

4.7 Exploitation of technical limits

A virus writer could analyse in detail how a scanner actually does the scanning and develop infection methods that cause detection problems for a specific scanner. The virus doesn't have to be difficult to find - it is enough that it would be very slow to search for it.

The Command Bomber virus is an example of this: it inserts it's code in the middle of the host file and builds a complicated series on branching commands to transfer the flow of the program code to the actual code. The detection of such virus would force some scanners to scan the whole file from the beginning to the end - which would be enough to make them unusably slow.

5. Attacks against resident scanners and behaviour blockers

Resident anti-virus programs are vulnerable to special attacks. Since DOS does not provide any kind of memory protection, a program can modify the memory space of another program. This makes it possible for a virus to locate and patch or disable a resident scanner or a behaviour blocker.

5.1 Unloading the protection

Some anti-virus TSRs can be unloaded from memory (actually, they will have to be unloadable if the product is wanted to be Novell-certified). If such mechanism exists, they can be called by a virus. This method is quite successful and well-known with some products.

5.2 Through the back door

Practically every TSR scanner has a back door, which is used by the non-resident scanner of the same package. This back door either turns off the checking done by the TSR or provides an alternative access method to the filesystem. If such back door would not exist, the TSR part would clash with the normal scanner, as the TSR would notice an infection when the non-resident part would open an infected file for scanning.

A virus could use such back doors for its own benefit, either disabling the resident part or by using the clean path to filesystem provided by the TSR.

Yet another way for a virus to attack a resident scanner would be to observe the display routines, and trap the alarm messages displayed by it. If the user never sees the alarm messages of the TSR, the protection is not doing its job.

6. Attacks against disinfectors

A retrovirus can attack programs that try to disinfect boot sectors and files. The purpose of such an attack might be to cause the disinfector to damage the host files while disinfecting. If a disinfection program does not do an exact identification on the virus before disinfecting, any virus that contains a known search string for another virus might cause such damage during disinfection process.

6.1 Cleaning the clean

There even exists a virus called Mirror, which is the exact opposite of a stealth-virus: it makes all programs look like infected by itself when the virus is resident in memory. This could be potentially dangerous when disinfection is attempted, but this technique poses no danger if the disinfection is done in a proper way, ie. after a clean boot.

6.2 Complicating the recovery

The recovery process of an infected machine can be severely complicated if the virus would deny access to the hard drive. Several MBR-viruses (for example, members of the Monkey family) do this by modifying the partition data in such a way that no logical DOS drives exist when the machine is booted from a clean floppy. A recovery done by overwriting the MBR code with the FDISK /MBR or similar command would not regain access to the hard drive.

The ExeBug virus family uses another way to make it difficult to boot up an infected machine from a clean diskette. The virus modifies the BIOS Setup information to indicate that the machine does not have A: drive at all. Such machine will always boot up from the hard drive. Once the booting has started and the virus code is executed, the virus will check if there is a diskette in drive A: and if so, it will continue the booting from there. In most cases the user is unable to notice this, and thinks that the machine has been booted clean when the virus is already resident.

Yet another way to complicate the recovery process would be to set the BIOS boot-up password on with a random password during an activation routine. The method of doing this is documented on most new BIOS brands.

Some integrity checkers are capable of performing a generic disinfection. This means that they try to restore the original file according to the information the checker has saved on it (typically length, checksum, first and last bytes). Such generic routine is unable to work if the virus makes extensive changes to the program files, for example, if the host file is encrypted during infection.

6.3 Attacking heuristic cleaners

A different kind of attack against disinfection programs is related to heuristic cleaners. A heuristic cleaner works by loading the infected file to memory and emulating the program code. It uses a combination of disassembly, emulation and sometimes execution to trace the flow of the virus and to emulate what the virus is normally doing. When the virus restores the original first instructions of the host file and jumps back to the original entry point, the cleaner stops the emulation. The repaired start of the program is copied back to the program file on disk and the part of the program that gained 'execution' will be removed. [Veldman]

The risk in heuristic cleaning is that if the cleaner tries to emulate everything the virus might get control inside the emulated environment and finally escape from it - after which it can propagate further or trigger a destructive retaliation routine. There are documented cases of at least one virus doing this, see below.

7. Attacks against integrity checkers

The operation of integrity checking programs varies between vendors but they almost always rely upon some form of a database which contains details of objects (typically files and boot sectors) to be checked.

7.1 Deleting the database

Several viruses have attacked integrity checkers by locating the integrity database and deleting it. In some cases the result of deleting the database files is that the integrity checker will blindly assume that the original checksums have not been calculated yet, and proceeds to initialize the database without informing the user that something might be amiss. This was exactly the case with the Peach virus.

Peach attacked an integrity checker which worked by creating a checksum file, containing checksums of all executable programs. Peach attacked by deleting this file. After the database was deleted and the checker was executed again, it would recreate the file, calculating new checksums from the infected files and fail to indicate any change in the system [VB1].

It should be noted that the Peach virus will not be successful against newer versions of this integrity checker, as the name of the checksum file has been changed in newer versions of the product. Similar types of attack still seem to be possible, though.

Even if a checksumming package does alert the user that the database has been deleted without approval, it would be difficult to find the affected files if no recent backup of the database exists.

7.2 Making checked unchecked

A similar attack works also against programs that do not store the integrity data to a separate database, but add it to the end of the executable files themselves. Since there is no info about which files have been checksummed, a virus can just remove the validation data without any side effects - and the checker will not complain that the file has changed.

Several generic attack methods against integrity checkers are discussed in length in [Bontchev].

8. Real world retroviruses

When looking at viruses that attack directly against specific anti-virus products, the most targeted ones seem to be McAfee Associate's ViruScan (SCAN.EXE), Microsoft Anti-virus from MS-DOS 6 (MSAV.EXE), Central Point Antivirus (CPAV.EXE) and the resident parts of these applications (VSHIELD and VSAFE). This is not surprising, as these are some of the most popular anti-virus products, and thus good targets for retroviruses.

Here are some examples of known viruses that incorporate retro-routines:

CPW virus family:

- tries to delete programs called TOOLKIT, GUARD, CHKVIRUS, SCAN, CLEAN, CPAV and VSAFE
- deletes CHKLIST.CPS files created by CPAV

Cybertech:

- deletes CHKLIST.CPS files
- removes the validation information added by SCAN and CPAV

Firefly:

- uninstalls VSAFE from CPAV or MSAV
- contains a segment of nested loops to confuse F-PROT's heuristic scanning
- deletes files called IM, VIRX, PCRX, VIRSTOP, MSAV, NAV, SCAN, CLEAN, TBAV, TBCSCAN, TBCLEAN, TBCHECK, TBMEM, TBSCANX, TBFILE, VC, and VCHECK

Goldbug:

- by-passes VSAFE.COM and DISKMON.EXE
- deletes or stops the execution of programs called SCAN, CLEAN, NETSCAN, CPAV, MSAV,
- TNTAV - and deletes the contents of CMOS memory at the same
- time specifically by-passes the TBAV boot-sector check deletes

CHKLIST.* files, by-passing CPAV and MSAV Lemming:

- disables TBDriver from TBAV by patching it in memory
- when TBScan is executed, adds the command-line parameter 'co', which will allow the stealth routines of the virus to operate
- patches text strings inside TBScan's code to make the operation of the program look like it

has been started without the 'co' switch Lockjaw virus family:

- deletes F-PROT, SCAN, IM, CPAV uninstalls VSAFE
- MtE.Groove and MtE.Encroacher:
- tries to delete files belonging to following products: Central Point Anti-Virus, Certus Novi, Fifth Generation Systems Untouchable, Norton Anti-Virus, Dr. Solomon's Antivirus Toolkit and VDS Virus Secure.

November_17th.890:

- overwrites the first 256 sectors of first hard disk whenever SCAN is run Peach:
- deletes CHKLIST.CPS files Sandra:
- tries to delete files belonging to CPAV, NAV, Untouchable, Dr. Solomon's Antivirus Toolkit and Integrity Master
- will not infect when FluShot is installed

Satanbug:

- tries to remove the validation codes added by SCAN
- guards it's own are-you-there interrupt call to make it difficult to detect the virus in

memory with it [CM-Base] Tequila:

- deletes files that have validation codes added by SCAN does
- not infect EXE-files which have the letters SC or V in name

Tremor:

- hooks INT 13h via a VSAFE back-door
- modifies it's own memory allocation when F-PROT is executed [VB2]

Varicella:

- tries to escape and go resident during the cleaning process of TBClean

9. Is there a real problem with retroviruses?

Do retroviruses pose a realistic threat against current anti-virus products? The most popular anti-virus tool is a stand-alone scanner, which by itself is almost always helpless against any new virus. Are there any special risks in a virus that, in addition to being a new one, also specifically tries to by-pass a product?

9.1 Dangers of optimized virus analysis systems

If a retrovirus exploits a specific flaw or back door in a product, it cannot be considered a very special case, as a new virus requires usually an update to the product anyway. At the same time it is possible to upgrade the product so that the attack-method used by the virus can be circumvented or made obsolete.

The main problem in this case is whether the anti-virus vendor notices what the virus is trying to do. Today, when several new viruses are found every day, there is limited time to analyse any single virus. Virus analysis systems are automated as much as possible, and a virus typically only gets a cursory look - which is usually enough to add detection, identification and disinfection. Such analysis will not reveal any special features the virus might contain. This also explains why there is no antivirus product that could provide detailed information about every virus.

If a retrovirus is run through a standard analysis system, and the product is tested by running it against a sample that is not resident in memory, the retro-features of a virus might not become known until they are featured in the real world - after which the virus will certainly get more attention, but this might already be a bit too late. The virus might also start its attack behaviours only after a certain latency time.

9.2 Opening the door to other viruses

It should also be noted, that a virus which disables an anti-virus product in some way could also be making the system vulnerable to other viruses, which the product would otherwise have handled fine.

In many cases this is the only benefit a retrovirus gains from unloading a resident scanner. The scanner can't be unloaded before it is resident. Once the scanner is resident, it will not let the virus run, if it is known to the scanning engine. If the virus is unknown to the scanner, it could have operated even when the resident scanner is active. The case is different with behaviour blockers, as they are not trying to find known viruses.

There is very little a product can do against an attack which consists of deleting or replacing the program file itself - if the virus gets control before the anti-virus, the virus makes the rules.

10. How should an antivirus product protect itself?

It is obvious that viruses can utilize a variety of tricks against anti-virus products. However, anti-virus programs can fight back just as efficiently.

10.1 Making the program difficult to locate

First of all, the anti-virus program itself should be renameable by the user. This alone would make it a lot harder for a virus to locate it's enemy. Unfortunately many anti-virus products refuse to run if they find that their program file has been renamed.

As the virus can try to locate the anti-virus program by it's contents as well as by name, the structure or contents of the program file should change with each update.

The best way to make sure that no retrovirus is making its tricks is the old, well-known recipe: boot from a clean diskette and run a fresh copy of the anti-virus program from diskette.

10.2 Self-checks

Since many attack-routines are based on modification of an anti-virus program, it is imperative that any anti-virus program should make thorough checks on its own code. A cursory check against modifications that would result from an infection is not enough: If the code is not protected internally against patching, the integrity of the whole program code should be checked during start-up.

It is not enough to ensure that the program code has not been changed. As demonstrated earlier in this paper, it is enough for a retrovirus to modify the texts or configuration info belonging to the application.

Even though the size of an anti-virus application probably changes during every update, a clever retrovirus could still locate the code it wants to patch by using a search string. This can be overcame by encrypting the application. The protection would be even better, if the encryption method or key would change with every update. Another, easier way to achieve the same results is to provide the executable in packed form, as the packing algorithm will invalidate search strings between different versions of the same program.

10.3 Resident security

Since it is often much easier to patch a program in memory rather than on disk, an antivirus application should make checksum checks on its memory image to ensure that no unwanted changes have taken place. This is important especially with the resident anti-virus utilities.

The communication channels to a resident part of the anti-virus program should be carefully thought out. If the TSR needs to have an uninstallation routine, it should be implemented so that it is difficult to have another program request for the uninstallation without the user noticing it.

10.4 Prohibiting disassembly

It can be expected that determined virus writers will try to disassemble anti-virus products in order to find out what makes them tick. Thus, some anti-debug and armouring code to protect the application might be a good idea - although nothing will stop a dedicated cracker.

At least three different scanners are known to have been analysed by crackers, up to the point of extracting all search strings of the program. Such attack can be harmful in several ways: the virus writers get to see exactly what they will have to change in a virus to make a new undetectable variant, and well-chosen search strings are also closely guarded trade secrets.

Popular, easy-to-get programs are the most probable targets for attack routines. This makes the commercial products theoretically more safe than shareware or freeware products.

11. Conclusions

Retroviruses are nothing new - the first ones were found in the late 1980's. There are several attack methods that will certainly be used in future viruses - and some of these can be quite efficient. Therefore, extreme care should be taken by producers of anti-virus software to avoid the possible pitfalls.

It's time to make sure your anti-virus product is not vulnerable to an attack it could avoid.

References

[CM-Base] Virus Test Center, University of Hamburg, CM-Base v3.0, March 1994, Satanbug entry by Padgett Peterson

[Veldman] Frans Veldman, Combating Viruses Heuristically, Proceedings, 3rd International Virus Bulletin Conference, September 1993, pp. 67-76

[Bontchev] Vesselin Bontchev, Possible Virus Attacks Against Integrity Programs And How To Prevent Them, Proceedings, 2nd International Virus Bulletin Conference, September 1992, pp. 131-141

[VB1] Virus Bulletin, Peach Virus Targets Central Point, Virus Bulletin May 1992, pp. 17-18

[VB2] Virus Bulletin, Tremor - A Shaky Start for DOS 6?, Virus Bulletin March 1993, pp. 10-11

[Fellows] F-Secure Ltd, F-PROT Professional User Guide rev 4.21

[Siilasmaa] Risto Siilasmaa, Building a Corporate Security Strategy - Coping With Computer Viruses, Proceedings, Cope'IT Conference 1993

BIOGRAPHY

Mikko Hypponen is a graduate from the Institute of Information Technology of Helsinki, Finland. He entered the antivirus field when he switched from being a database developer to a full-time virus specialist in 1990.

Hypponen works as the Support Manager at F-Secure Ltd.

Converted to html in May 1996, MHH